CodeBlocks 手册 使用篇

译者：JGood

http://blog.csdn.net/JGood/archive/2010/01/25/5252119.aspx

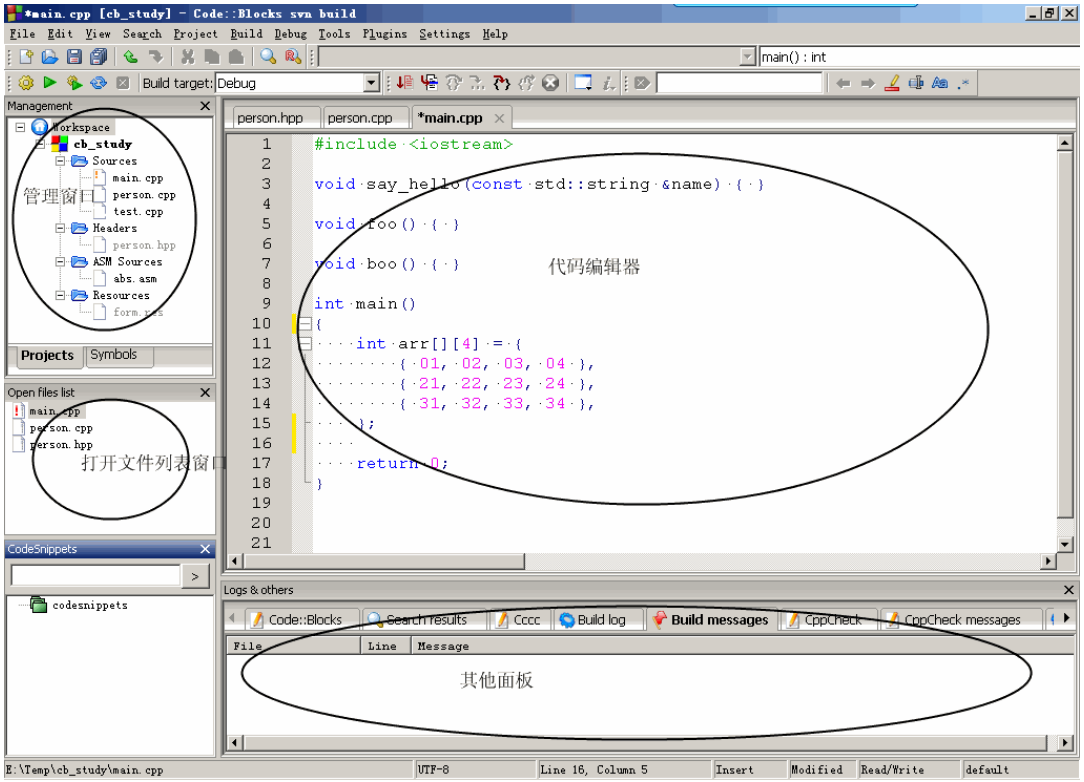
原手册：http://www.codeblocks.org/docs/manual_en.pdf

笔者打算用两篇日志来完成对 Code::Blocks 手册前二章的编译，分别是：使用篇、插件篇。本文是第一篇：Code::Blocks 使用篇 。原手册第三章介绍 Code::Blocks 变量、脚本的使用，第四章介绍如何从源码编译 Code::Blocks，这两章内容不是很多，笔者认为对大部分用户帮助不是不大，暂不打算翻译。笔者使用的 Code::Block 版本是 nightly builds，svn6088（可以在这个地址下载：http://forums.codeblocks.org/index.php/topic,11875.0.html ）。使用的编译器是 GCC3.4.5。每个版本之间的使用可能会有细微的差别。

Code::Blocks 手册 Version 1.0

感谢 CodeBlocks 项目组：

Anders F. Bjorklund (afb), Biplab Kumar Modak (biplab), Bartomiej wiecki (byo), PaulA. Jimenez (ceniza), Koa Chong Gee (cyberkoa), Daniel Orb (daniel2000), Lieven de Cock(killerbot), Yiannis Mandravellos (mandrav), Mispunt (mispunt), Martin Halle (morten-macy), Jens Lody (jens), Jerome Antoine (dje), Damien Moore (dmoore), Pecan Heber(pecan), Ricardo Garcia (rickg22), Thomas Denk (thomasdenk), tiwag (tiwag)



1 CodeBlocks 项目管理

1.1 基本视图

下图是 CodeBlocks 运行时的用户界面：

管理(Management)**:** 管理窗口包含 Project 视图与 Symbols 视图。Project 视图显示当前 CodeBlocks 打开的所有项目（译者注：类似与 VS 解决方案资源管理器）；Symbols 视图显示项目中的标识符：类，函数、变量等信息（译者注：类似与 VS 的类视图）。

代码编辑器：支持代码折叠，关键字高亮显示。上图 main.cpp 正在被编辑。

打开文件列表：显示当前在代码编辑器中打开的所有文件列表。上图中打开的文件列表为：main.cpp, person.cpp, person.hpp

代码段（CodeSnippets）：管理常用的代码段、常用文件链接(links to files)与 URL。可以通过菜单 View->CodeSnippets 来显示该面板。

日志和其他：这个窗口用于输出日志信息，显示查询结果等等。

状态栏提供了以下这些信息：

编辑器中打开文件的绝对路径；

文件的编码类型；

光标所在的行与列；

当前的键盘模式（insert 或者 overwrite）；

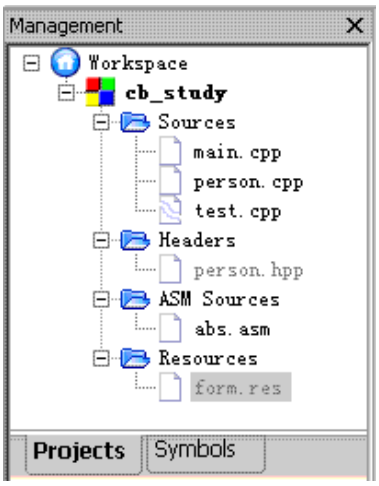
当前的文件状态。被修改过的（但尚未保存）文件将被标记为“modified”，否则这里为空；

文件操作的权限。如果文件是只读的，这里将会显示“Read only”，在 Open files list 中，该文件会使用一个加琐的图标来显示；

个性化配置名称；

CodeBlocks 提示了非常灵活和强大的项目管理功能。下文将介绍项目管理的一些常用功能。

项目视图（Project View）



在 CodeBlocks 中，Project 的源文件（如 C/C++的源文件及其对应的头文件）和编译属性设置信息都保存在<name>.cbp 文件里。可以通过菜单[File－>Project]启动工程创建向导来创建新的 Project，然后通过管理窗口的上下文菜单[Add files]（译者注：上下文菜单，指当前窗口中选中目标项目，然

后右键显示的菜单）向 Project 中添加文件。CodeBlocks 会自动根据文件的后缀将它们放入不同的类别文件夹中（译者注：这个文件夹是虚拟的，实际并不存在）。下面是默认的分类：

Sources：包含源文件，后缀为*.c、*.cpp；

ASM Sources：包括汇编源文件，后缀一般为*.s、*.S、*.ss、*.asm；

Headers：包括头文件，后缀一般为*.h、 *.hpp；

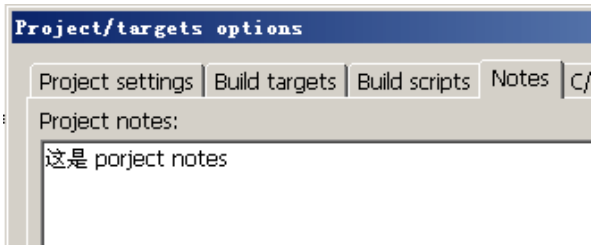
Resources：包括资源文件，后缀一般为*.res、*.xrc；

通过管理窗口的上下文菜单[Project tree －>Edit file types & categories]，用户可以新建自定义的文件类别，并为其相应的后缀。例如，如果你希望*.ld 的文件放到 Linkerscript 的分类里，你只需新建类别 Linkerscript，并为它指定*.ld 后缀即可。

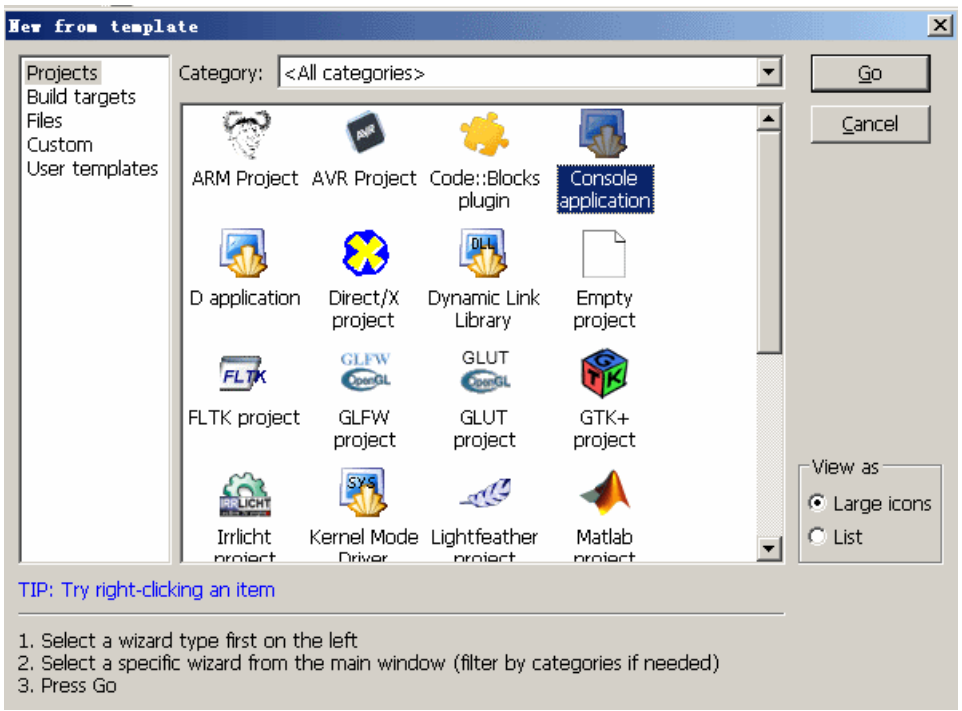
提示： 如果你取消选中管理窗口上下文菜单的[Project tree－>Categorize by file types]选项，所有的项目文件将会按它们在文件系统中的位置来显示。

1.2 项目备注（Notes for Projects）

可以给 CodeBlocks 项目添加一些备注，用于对项目进行概要的描述与说明，这些信息有助于其他成员迅速的了解项目。备注信息被保存在项目工程文件里，并可以设置为随着项目的启动而显示。如图：

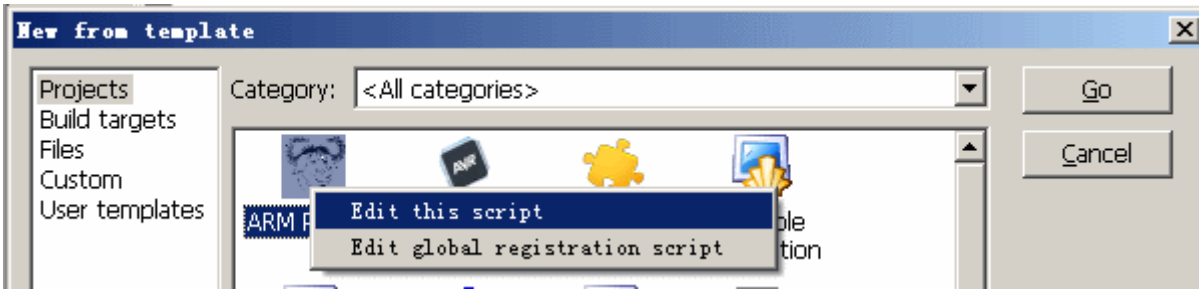


1.3 项目模板



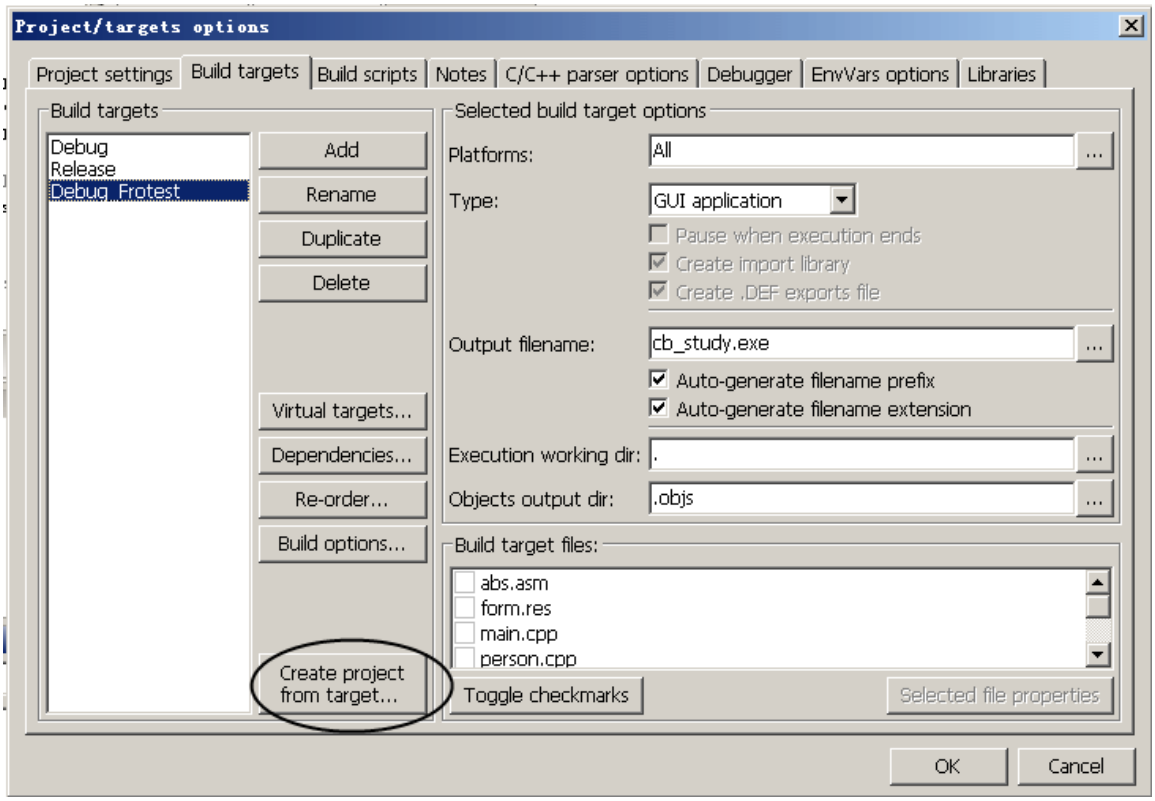
CodeBlocks 支持许多不同类型的项目模板，它们会在新建项目的时候显示，创建新项目的时候往往从这些模板中选择（如上图：）。用户可以自定义工程模板。工程模板保存编译器的类型、编译选项、资源的配置等相关信息。项目模板保存在 Documents andSettingsn\<user>\Application Data\codeblocks\UserTemplates 目录中。如果你希望该工程模板被本机的所有用户使用，必须把对应的模板文件拷贝到 CodeBlocks 的安装目录。模板在 CodeBlocks 重启之后生效，通过菜单[New->Project->User templates]就可以看到自定义的模板。

提示： 用户可以通过右键选中工程向导中的模板对其进行编辑。如下图：



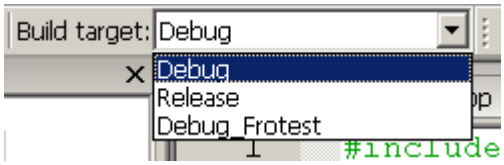
1.4 从编译模式创建项目（Create Projects from Build Targets）

一个项目往往有不同的编译模式，最常见的是 Release 和 Debug。不同的编译模式表示用于构建当前项目所使用的不同编译器选项、调试信息或者依赖的文件。每一种编译模式都可以被保存成独立的工程文件。通过上下文菜单[Project->Properties]，在 Build Targets 标签中点击“Create project from target”按钮来生成对应编译模式的工程文件，如图：



1.5 编译模式（Virtual Targets）

一个项目可以有多种不同的编译模式，最常用的编译模式为“Debug”和“Release”。“Debug”模式下会包含许多 Debug 信息，而“Release”模式下没有这些信息。也可以通过上下文菜单[Project->Properties->Build Targets]添加其他的编译模式。编译模式将显示在工具栏中：



1.6 预生成和生成后步骤 (Pre- and Postbuild setps)

Code::blocks 允许在项目生成前和生成后执行额外的操作，这些操作分别被称为预生成（Prebuilt）或生成后（Postbuilt）步骤。下面是典型的生成后步骤：（译者注：笔者对低层的东西知道的不多，不是很清楚下面这些步骤的意思。）

Creating an Intel Hexformat from a nished object

Manipulating objects by objcopy

Generating dump les by objdump

1.7 为构建目标添加脚本（Adding Scripts in Build Targets）

Code::Blocks 允许使用脚本来调用菜单命令，控制项目的生成。

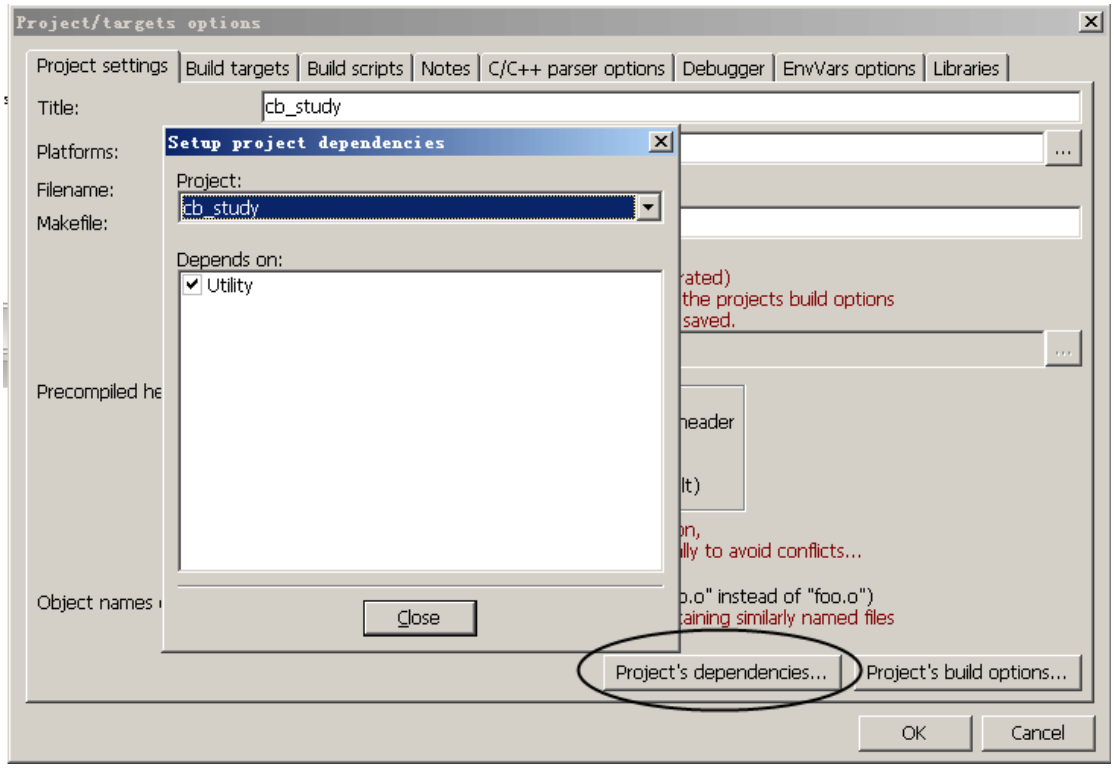
提示： 脚本可以被包含到一个构建目标（Build Target）中。

1.8 解决方案（Workspace）与项目依赖

CodeBlocks 可以同时打开多个项目，通过菜单[File->Save workspace]把它们集成到一个解决方案中，并生成一个对应的解决方案文件（<name>.workspace）。下次打开解析方案文件(<name>.workspace)时，这些项目会被一起加载进来。

复杂的软件系统通常由不同的模块、组件以独立的工程项目组成，它们之间往往存在依赖关系。

例如：项目 A 以库(library)的形式提供一些公用函数给其他项目调用，如果项目 A 中的源代码被修改，这个库就得重新编译。项目 B 使用项目 A 中实现的函数，那么项目 B 就依赖项目 A。CodeBlocks 把这些项目依赖的信息保存到解决方案文件中，所以解决方案中的工程可以各自独立的创建、编译而不相互影响（译者注：由解决方案文件来维护各项目的依赖关系）。这些依赖关系会决定项目的生成顺序。通过菜单[Project->Properties]，然后选择[Project’s dependencies]按钮来设置项目之间的依赖关系。如下图：



1.9 包含汇编文件

略。

1.10 代码编辑器与工具 (Editor and Tools)

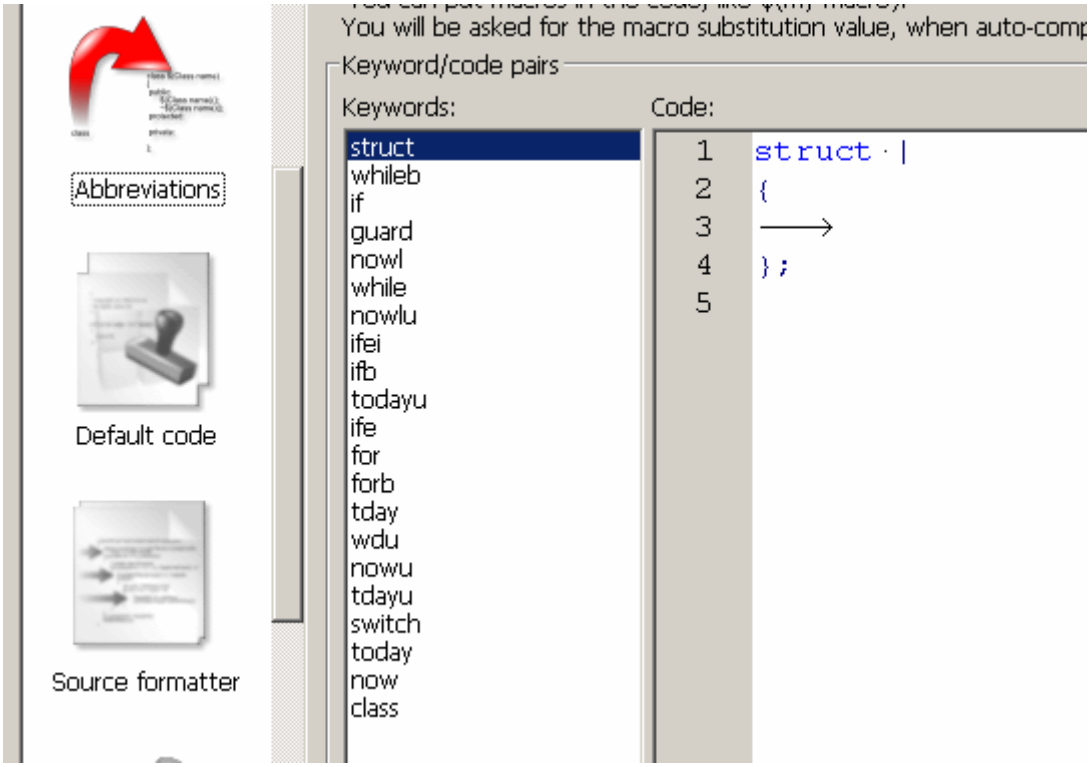
1.10.1 默认代码 (Default Code)

公司的编码规范往往要求源文件有统一的布局（译者注：例如源文件的开始处以统一的格式给出文件创建的时间、作者、实现的功能描述等相关信息）。CodeBlocks 允许预定义一些内容，当新建C/C++文件时这些内容会自动添加到文件的开始处。这里把预定义的内容称为 default code。可以通过菜单[Settings－>Editor－>Default Code]来设置 defaulg code，通过菜单[File->New->File]创建的代码文件将自动添加上 default code。例如：

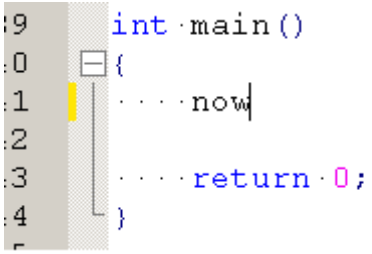
```
/*************************************************************************
* Project:
* Function:
*************************************************************************
* $Author: mario $
* $Name: $
*************************************************************************
*
* Copyright 2007 by company name
*
*************************************************************************/
```

1.10.2 缩写（Abbreviation）

定义一个常用的代码片断[typing]，并给它提供一个名字[Abbreviation]，在写程序的时候，只要给出这个名字，然后按快捷键 Ctrl + J，CodeBlocks 就会用预先定义的代码片断来替换该名字。（译者注：VS 中也有类似的功能，在 C#程序时，只要写出关键字 for，然后连续按两次 Tab 键，编辑器会自动生成 for 语句的框架，是不是很方便？~_~）。通过菜单[Settings->Editor]来设置 Abbreviation，如下图：

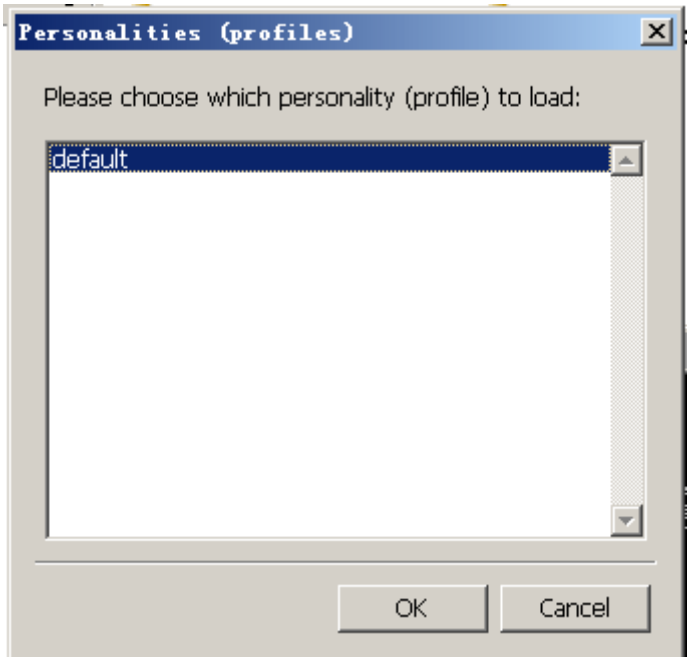


abbreviation 也支持参数变量(Parametrisation)（如：$NOW 表示当前时间）。如：



1.10.3 个性化（Personalities）

CodeBlocks 的配置信息被作为应用程序数据而保存在 codeblocks 文件夹的<user>.conf 文件中。该配置文件中保存的信息包括：上次打开的项目、代码编辑器的设置、符号栏显示等等。默认的个性化设置被保存在 default.conf 文件中。通过命令行参数-personality=myuser 来调用 CodeBlocks，配置信息将被保存到 myuser.conf 中。如果该文件不存在，系统将自动创建它。如果以命令行的方式来启动 CodeBlocks，并传递命令参数--personality=ask，将会显列出当前所有的修改化配置列表，用户选择其一启动 CodeBlocks。如下图：



1.10.4 配置文件（Configuration Files）

CodeBlocks 的配置信息保存在 codeblocks 目录下的 default.conf 文件中。当使用个性化设置的时候，配置信息将被保存到<personality>.conf 文件里。

cb_share_conf（一个辅助工具，可以在 CodeBlocks 安装目录里找到）被用来管理与保存这些设置信息。

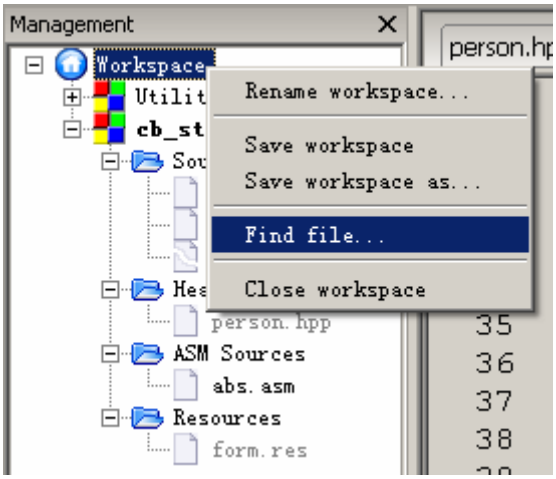
如果你想为电脑的不同账户定义一个公用的默认设置，defual.conf 文件应该被保存到\Documents and Settings\Default User\Application Data\codeblocks。CodeBlocks 在第一次启动的时候，会将该配置文件拷贝到当前账户的 application data 目录下（并作为当前帐户的配置文件）。

如果想在 usb 设备上创建一个绿色版本号的 CodeBlocks，请执行下面步骤：将 CodeBlocks 安装目录拷贝到 usb 设备上，将配置文件 default.conf 拷贝到该目录中，该配置文件将被用于全局的设置。确保系统有权限对该文件进行写入，否则 CodeBlocks 对配置文件的修改将不会被保存。

1.10.5 导航与搜索（Navigate and Search）

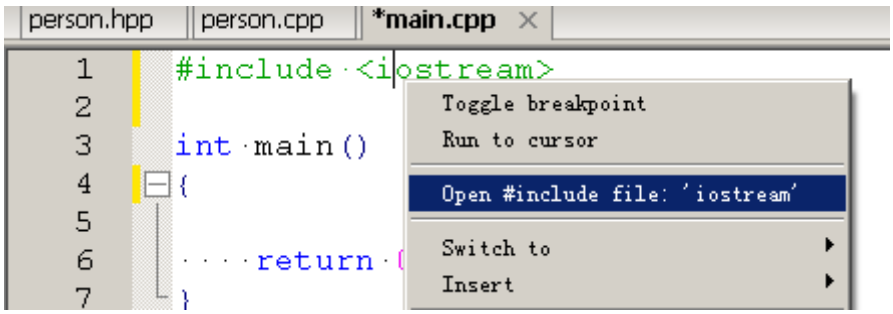
CodeBlocks 提供了很多方法用于文件和函数之间导航。书签就是最常用的一种方式。通过快捷键 Ctrl + B 在源文件中设置/删除一个书签，通过 Alt + PgUp 或 Alt + PgDn 在不同的书签之间跳转。

在管理窗口的 Project 视图中选中解决方案或项目，右键点击在弹出菜单中选择[Find file]，输入你查找的文件名称，然后回车，该文件将被选中（如果文件存在的话），再按回车，编辑器就会打开该文件。如下图：



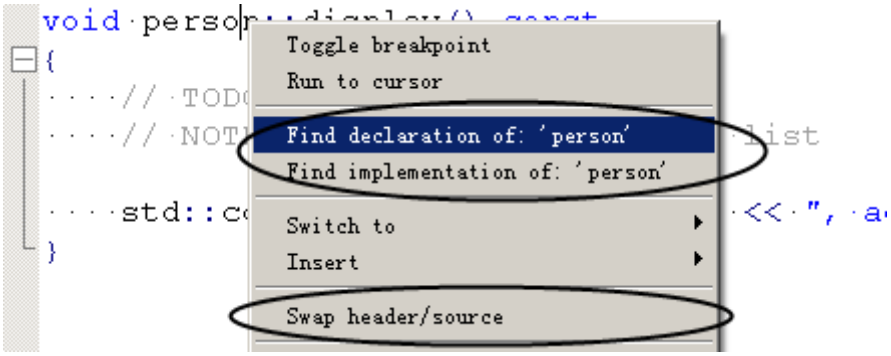
在 CodeBlocks 中，你可以很容易的在头文件与源文件之间导航：

1. 将光标置于头文件被包含处（例如：#include “header.hpp”），右键选择性“open include file”，编辑器将打开该包含文件。（译者注：可以在 VS 中使用快捷键 Ctrl + Shift + G 实现同样的操作。这个功能非常方便，特别是要查看源代码时。）



2. 通过右键菜单的[Swap header/source]，在头文件与源文件之间切换。

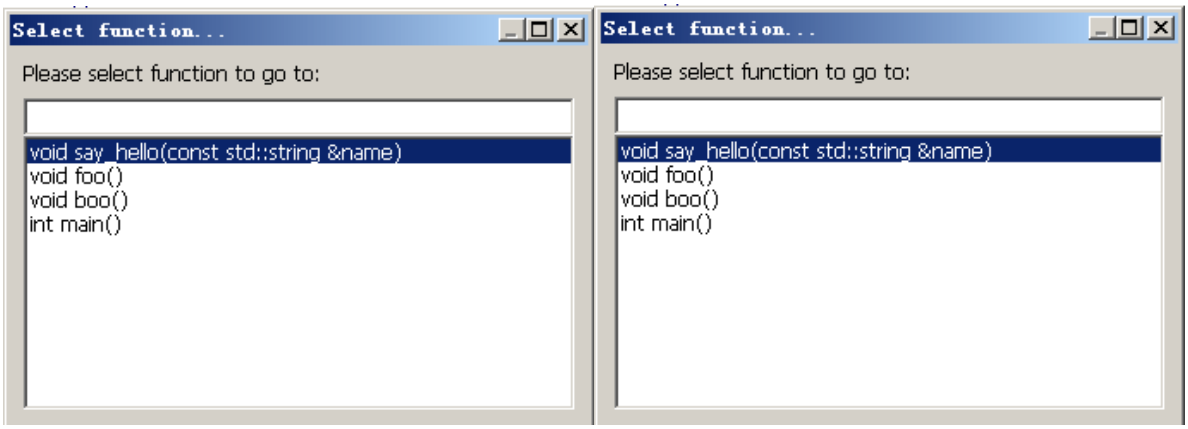
3. 选中一个定义（如变量名，类型名等等），在右键菜单中选择[Find declaration]，编辑器就会打开该定义的声明。（译者注：可以使用右键菜单的[Find implementation]定位到函数的实现处，在 VS 中使用快捷键 F12 实现同样的操作。这也是一个非常方便的功能。）。



CodeBlocks 提供多种方式用于对单个文件或整个目录进行搜索。通过菜单[Search->Find]或[Search －> Find in Files]来打开搜索对话框。

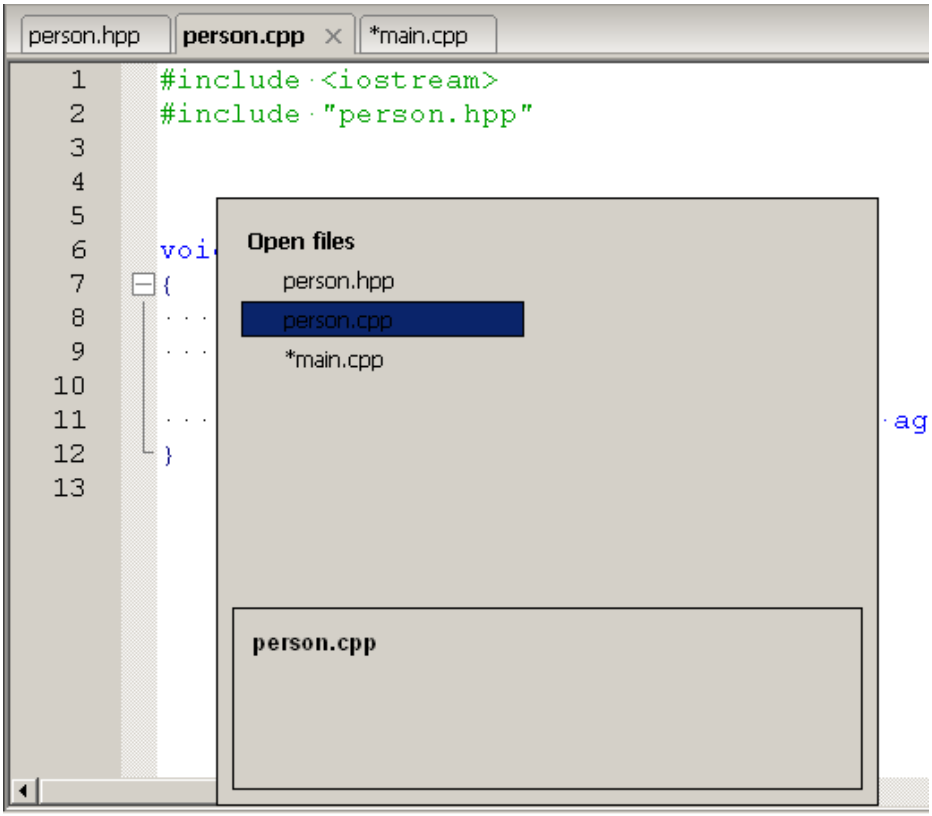
Alt + G 和 Ctrl + Alt + G 这两个快捷键用于打开 文件/函数 跳转对话框，只要输入文件/函数的名称，就可以很方便的跳转到指定文件或函数。文件名/函数名搜索还支持 * 和 ？ 等能配符。（译

者注：可以使用 Visual Assist 插件在 VS 中实现类似的功能。这两个功能真的很方便实用。）



提示： 使用 Ctrl + PgUp 和 Ctrl + PgDn 可以在前后函数之间跳转。

在文本编辑器中，使用快捷键 Ctrl + Tag 可以在当前所有打开的文件之间跳转。（译者注：VS 也有类似的功能，而且快捷键也是 Ctrl + Tag，这是巧合吗？）

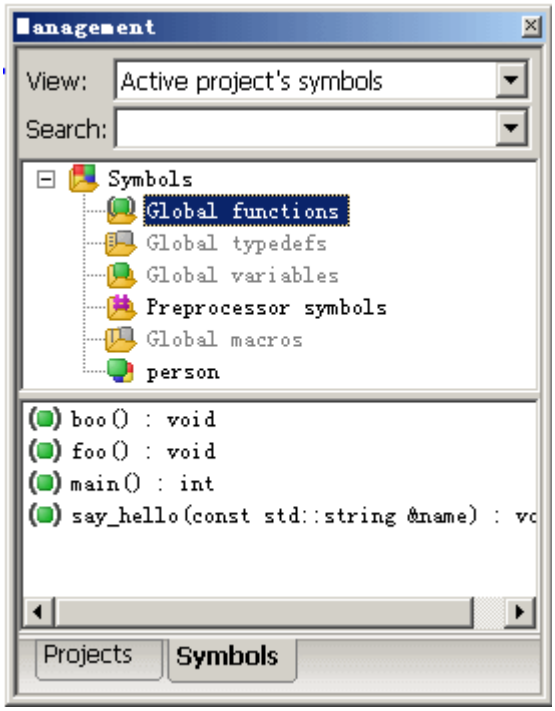


显示行号。通过菜单[Settings->General Settings]，选中[Show line numbers]来显示行号。使用快捷键 Ctrl + G 可以快速的跳转到指定行。

1.10.6 符号视图（Symbol view）

CodeBlocks 管理窗口提供的符号视图，以树的形式显示（导航）C/C++源文件中的类、函数、变量定义。可以选择符号显示的区域是：当前文件、当前项目、整个解决方案。（译者注：与 VS 的

类视图面板实现类似的功能。）



提示： 在搜索输入框中输入符号的名称，符号浏览器将会过滤不符条件的符号。

符号视图将符号分为如下分类：

Global functions: 全局函数；

Global typedefs: 通过 typedef 定义的别名；

Global variables：全局变量；

Preprocessor symbols: 通过#define 宏定义的预处理指示符；

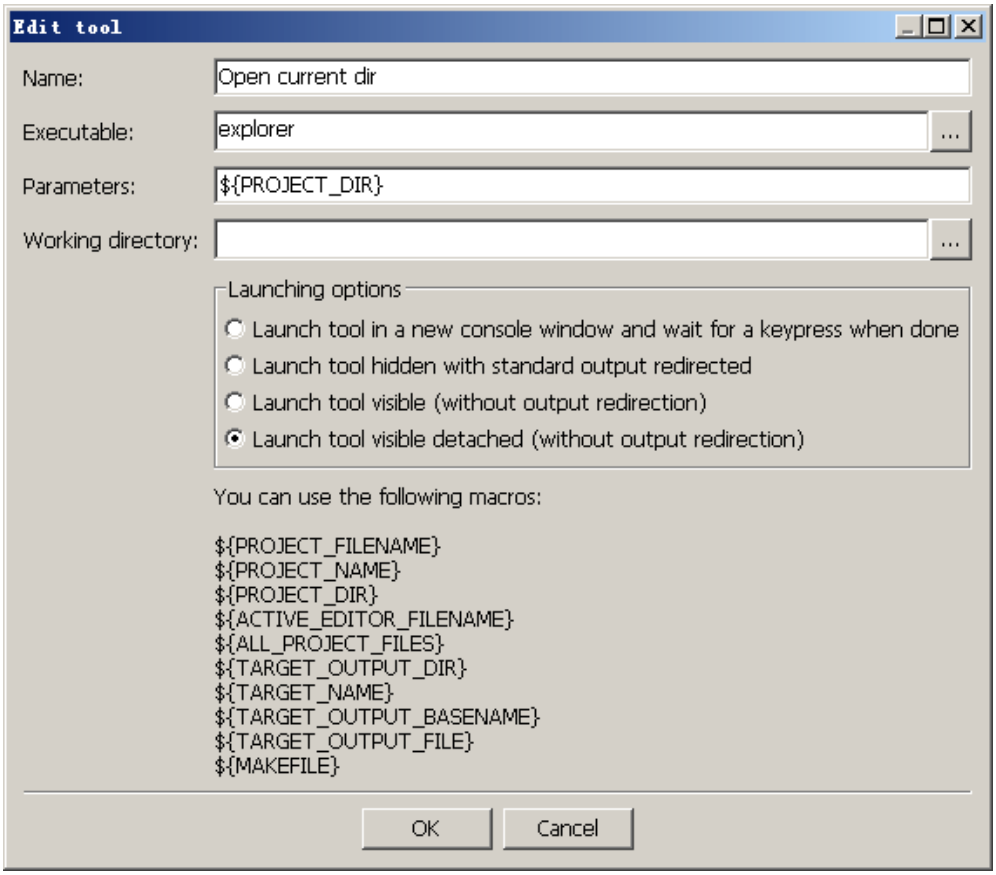
结构和类的定义显示在 pre-processor symbols 下面。如果一个分类被选中，属于该分类的标识符将显示在视图的下半部分。双击这些标识符，代码编辑器将定位到定义这些标识符的文件所在处。

1.10.7 集成外部的帮助文档

CodeBlocks 支持添加外部的帮助文档集成到开发环境中。通过菜单[Settings->Environment]来设置。把你选择的 chm 格式的文档以添加到 Help Files，并将其作为默认的帮助文档，在编辑器中选择一个函数，对应的文档就会出现通过快捷键 F1。…

1.10.8 集成外部工具

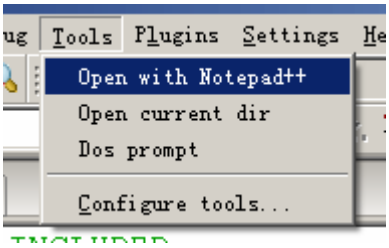
通过菜单[Tools->Configuration Tools－>Add]，把外部工具集成到 CodeBlocks 开发环境中。这些外部的工具允许以参数的形式访问 CodeBlocks 的内建(Built-in)变量（如当前项目所在的文件夹${PROJECT_DIR}）。利用这个功能，我们给 CodeBlocks 添加一个菜单项：打开当前项目所在的文件夹。请看图：



（译者注：这是一个非常实用的功能。利用这个功能，我在我的 CodeBlocks 中，添加了三个我最常用的菜单项：使用 Notepad++打开当前文件；打开当前项目所在的文件夹；以 Dos 窗口打开当前文件夹。爽歪歪~~）

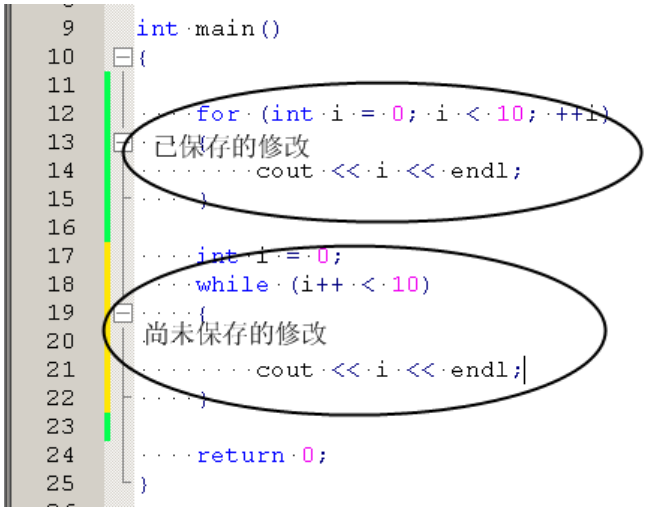
1.11 使用 CodeBlocks 的小提示（Tips）

这节我们将展示一些 CodeBlocks 非常有用的提示。



1.11.1 修改跟踪（Tracking of Modifications）

CodeBlocks 通过行号旁边的小竖条颜色来跟踪源文件的修改。未保存的修改行将被标记为黄色，而已保存的修改行标记为绿色。你可以通过菜单[Search->Goto next changed line]或者[Search->Goto previous changed line]在修改内容之间导航（对应的快捷键是 Ctrl + F3 和 Ctrl + Shift + F3）。（译者注：VS 也提供类型的功能。）



可以通过菜单[Settings->Editor->Margins and caret]，取消选中[Use Changebar]来取消该功能。

提示： 如果文件被关闭，记录在该文件上的 undo/redo 信息和修改标识(changebars)将会清空。如果文件处理打开状态，可以通过菜单[Edit->Clear changes history]或者右键菜单相应选项来显式地清空这些信息。

1.11.2 与其他应用程序交互

CodeBlocks 能够在运行时与其他应用程序进行交互。windows 下通过 DDE（Dynamic Data Exchange）实现这种进程间的交互，而在其他操作系统下，基于 TCP 来实现交互。

以下语法的命令可以发送给 CodeBlocks 运行实例：

[<command>(“<parameter>”)

当前可以使用的命令：

Open： 命令[Open(“D:\Temp\test.txt”)]，在 CodeBlocks 实例中（或者启动新的 CodeBlocks 进程，如果需要的话）打开一个文件。

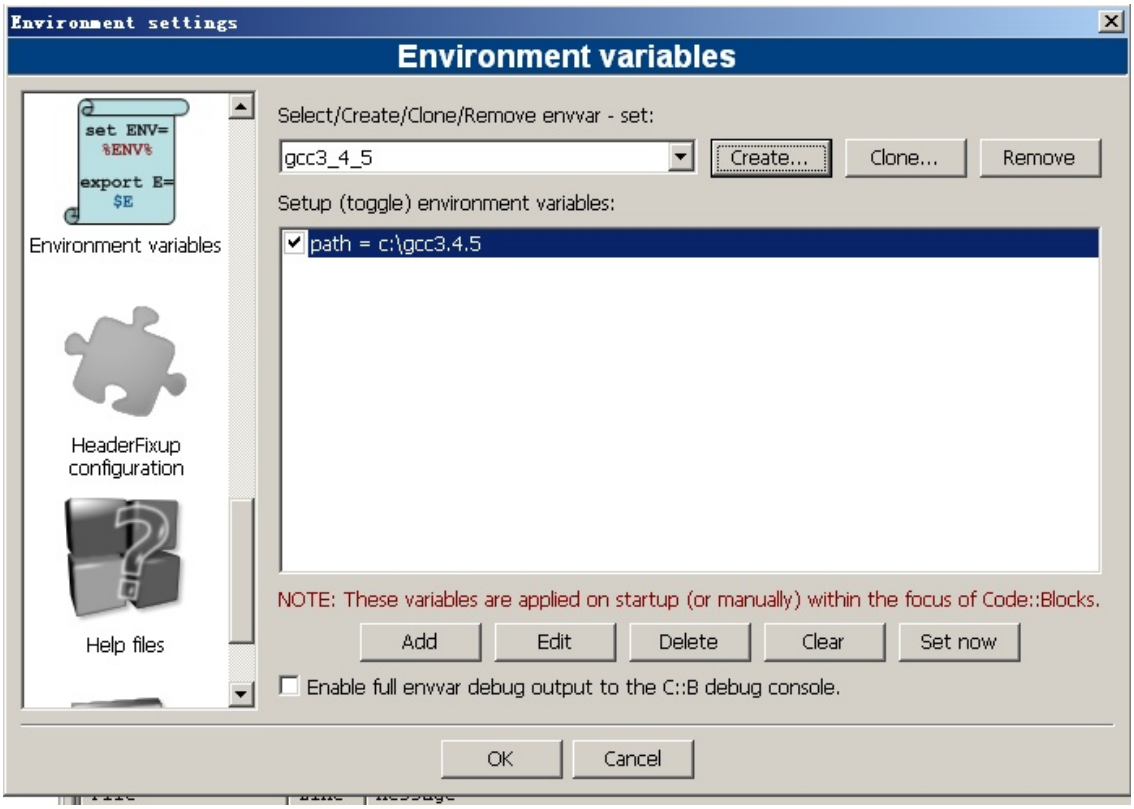
OpenLine：命令[OpenLine(“D:\Temp\test.txt:10”)]，在 CodeBlocks 中打开文件，并定位到指定行数，冒号后面的数字用于指定行号。（译者注：不是很明白原文的意思：This command opens a le at a given line number in a CodeBlocksi nstance.）

Raise：让 CodeBlocks 实例获得焦点。不就给该命令提供参数。

1.11.3 配置环境变量

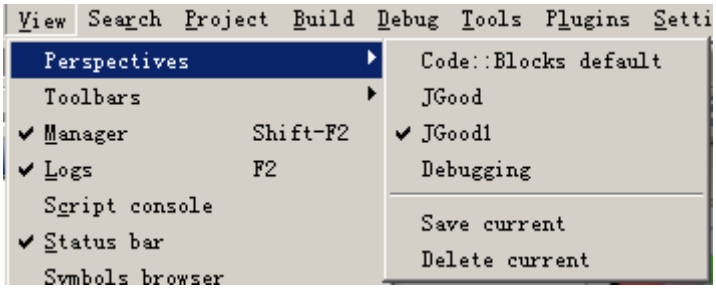
操作系统的配置信息被定义在环境变量中。例如环境变量 Path 包含一个编译器的安装目录路径，操作系统在运行期间都可以随时访问该环境变量。如果同一编译器的另一个版本被安装，就可能会发生错误，如：调用的编译器版本不正确。

有时候因为项目的需要，必须在同一机器上安装同一编译器的不同版本。为了避免上述错误的发生，可以在项目启动之前修改环境变量。显然，这个方法很容易出错，很不灵活。基于这个问题，CodeBlocks 提供了一个简单的解决方法：创建不同版本的、只能在 CodeBlocks 内使用的环境变量，然后根据项目的需要选择适当的版本环境变量。通过菜单[Settings->Environment]，在 Environment Varibales 面板中点击 Create 按钮创建不同版本的环境变量，如下图：



1.11.4 切换布局

CodeBlocks 可以根据手头任务的需要选择不同的配置/视图，并保存这些配置/视图。默认情况下这些配置保存在 default.conf 文件里。以命令行方式启动 Cdoeblocks，并传递--personality=ask 参数，就可以在列出的个性化配置列表中选择一个视图启动。有时候，你可能希望在应用程序的使用过程中切换布局，如编码模式与调试模式下可能需要不同的布局。CodeBlocks 提供了一种机制来实现这种需求，通过菜单[View->Layouts->Save current]，并输入布局的命名，保存布局。通过[View->Layouts->Layout name]来切换布局。（译者注：笔者使用的版本，通过[View->Perspectives]来保存\切换布局，如下图：）



1.11.5 项目切换

多个项目同时在 CodeBlocks 中打开时，用户希望快速的在这些项目之间切换。CodeBlocks 提供一组快捷键来实现：

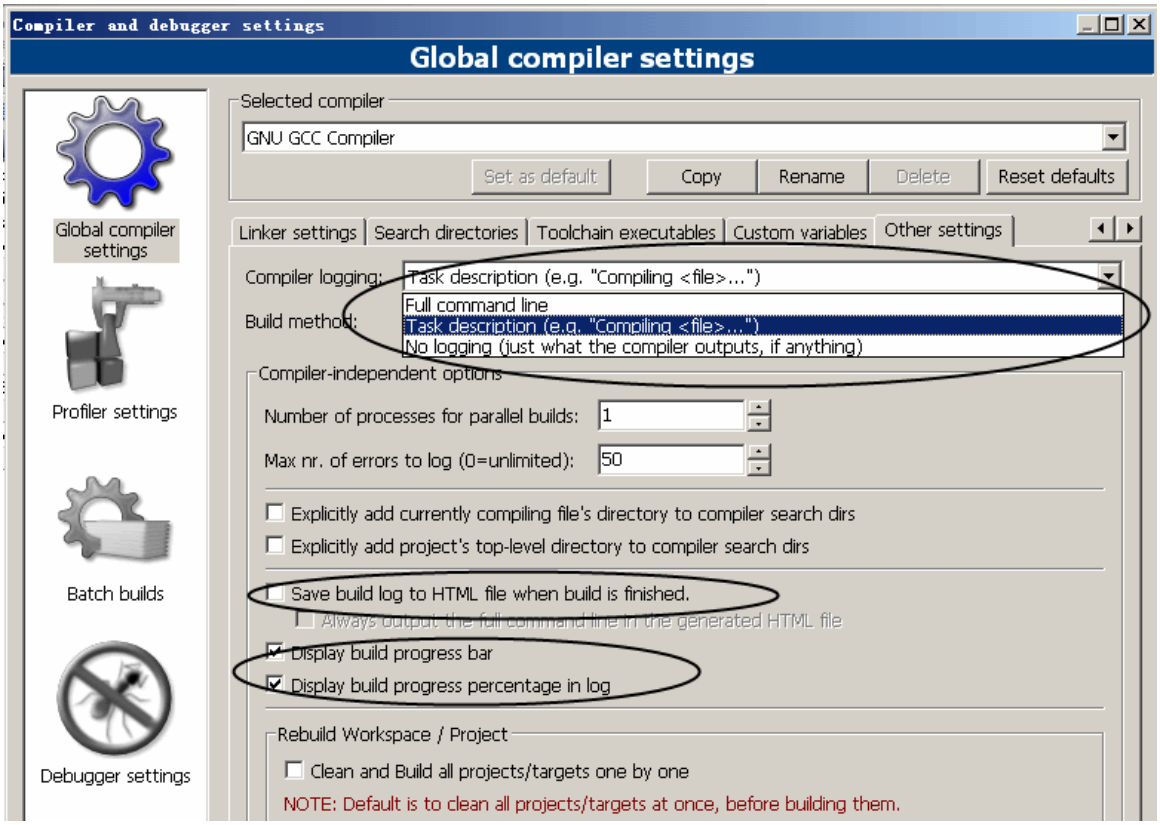
Alt + F5 ：将视图中前一个项目设为激活(Active)项目。（译者注：类似于 VS 中将某个项目设为启动项目。）

Alt + F6： 将视图中后一个项目设为激活项目。

F11： 在编辑器中切换源文件(name.cpp)和头文件(name.h)。

1.11.6 扩展编译器设置

在编译一个项目的过程中，编译信息将会显示在消息窗口的 Build Log 视图中。如果你想获取更详细的编译信息，通过菜单[Settings->Compiler and Debugger]，在 Other Settings 面板中选择 Compiler logging 下拉列表项：



“Full command line”选项意味着将在 Build Log 视图中显示所有的编译信息。可以将这些日志信息保存为 HTML 文件，通过选中”Save build log to HTML file when finished’。另外，通过选中”Display build process bar”，CodeBlocks 支持在 Build Log 视图中显示编译进度。

1.11.7 编辑器缩放

CodeBlocks 提供了一个非常高效的编辑器。这个编辑器允许你缩放打开文本的字体大小。如果你的鼠标有滚轮，你只要按住 Ctrl 键，同时滚支鼠标滚轮，就可以实现文本的缩放。

提示： 通过菜单[Edit->Special commands->Zoom->Reset]来重置缩放。

1.11.8 自动换行模式

在 CodeBlcoks 中打开一个文本文件，使用自动换行模式可以将很长的行以多行的形式显示在一个屏幕内，有利于编辑。通过菜单[Settings->Editor->Other Options]，选中 Word warp 来激活自动换行模式。…

1.11.9 块选文本

CodeBlocks 支持在代码编辑器中块选文本。按住 Alt 键的同时按住鼠标左键在编辑器选择一块区域。如果你想选择数组的几列进行复制和粘贴，这个功能是非常有用的（译者注：VS 也提示类似

的功能，快捷键也一样。)。如图：

```
    int arr[][4] = {
        { 01, 02, 03, 04 },
        { 21, 22, 23, 24 },
        { 31, 32, 33, 34 },
    };
    return 0;
```

1.11.10 代码折叠

CodeBlocks 支持代码折叠，允许将函数、类的实现折叠起来。

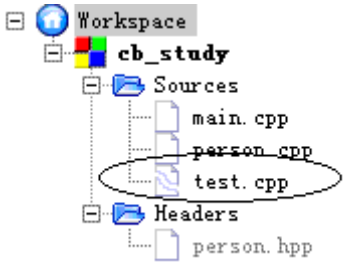
提示： 通过菜单[Settings->Editor->Folding]，可以设置代码折叠的样式和层次限制（depth limit）。

1.11.11 自动完成

在 CodeBlocks 中打开一个项目时，编译器的相关目录（include 目录）与项目的源文件/头文件将被解析，提取有关类型定义、函数、宏等的信息，CodeBlocks 使用这些信息来实现自动完成功能。通过菜单[Settings->Editors->Code completion]启用该功能，通过快捷键 Ctrl + Space 来显示提示信息（译者注：默认的快捷键 Ctrl + Space 不适合中国人的键盘习惯，建议改成其他快捷键。)。通过[Settings->Editor->Syntax highlighting]，你可以添加自定义的关键字。

1.11.12 查找破损文件（Find broken files）

如果一个文件已经从磁盘中删除，但它仍然被包含在项目文件(project.cbp)中（译者注：项目的文件信息保存在*.cbp 文件里。)，这个文件在项目面板中显示一个破损符号（如下图)。应该通过上下文菜单[Remove file from project]将它从项目中移除。



一个大的工程可能包含许多子文件夹，搜索破损文件会非常花费时间。CodeBlocks 提供 ThreadSearch 插件来解决这个问题。在 ThreadSearch 中输入要查找的表达式，并设置查找的范围：“Project files”或者“Workspace files”，ThreadSearch 将会分析所有包括在项目或者解决方案中的文件。当 ThreadSearch 找到一个跋损文件的时候，会发出一个文件丢失的错误。

1.11.13 包含库

在项目的编译选项中，你可以通过”Add”按键添加项目所使用的库。库可以以绝对路径的形式给出，也可以只给出名称而无需提供 lib 前缀和文件扩展名。

例如：存在这样的一个库文件：<path>\libs\lib<name>.a，只要给出<name>，链接器就可以找到对应的库文件。

1.11.14 对象链接顺序

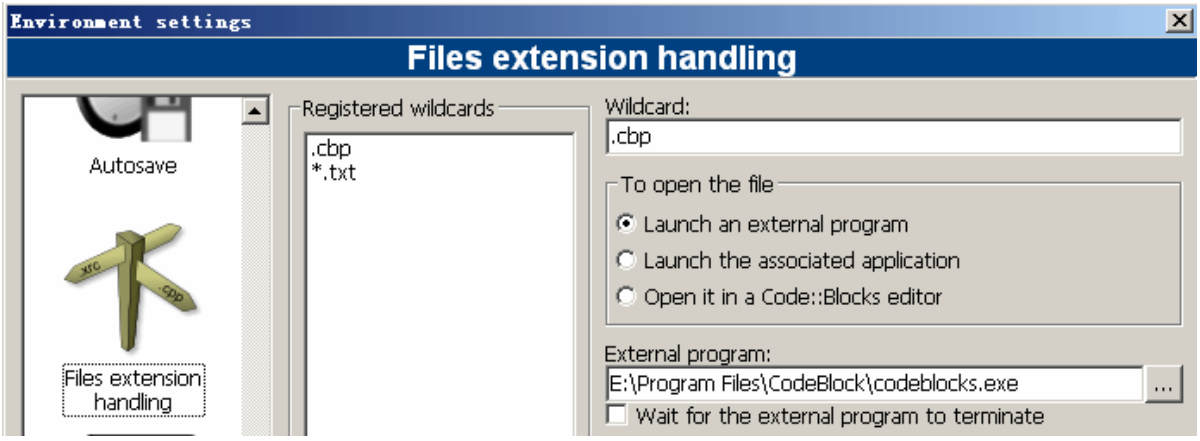
在编译过程中，源文件 name.c/cpp 将会被编译成目标文件 name.o，然后链接器把独立的目标文件链接成可执行文件 name.exe（对于嵌入式系统，将链接成 name.elf）。这种情况下，可能需要预先定义对象链接的顺序。在 CodeBlocks 中，可以设置相关源文件的优先级来实现。在上下文菜单中选择[Properties]，在 Build 标签中定义。较低优先级使文件较先链接。

1.11.15 自动保存

CodeBlcoks 允许自动保存当前正在编辑的文件和项目，或者对它们进行备份。可以通过菜单[Settings->Environment->Autosave]来激活该功能。

1.11.16 文件扩展名设置

在 CodeBlocks 中，可以选择多次方式来处理文件不同后缀的文件，可以在菜单[Settings->Environment settings->Files extension handling]设置相应的处理方式：“Launch an external program”指定外部程序来打开文件；“Launch the associated application”使用系统默认程序来打开文件；“Open it in Code::Blocks editor”使用 Code::Blocks 编辑器来打开文件。如下图：



1.12 通过命令行操作 CodeBlocks

CodeBlocks 能够通过命令行来运行。在这种情况下，需要通过一些选项来控制项目的构建过程。因为 CodeBlocks 是 scriptable 的，所以 CodeBlocks 项目的构建可以集成到你自己的工作过程中。

codeblocks.exe /na /nd --no-splash-screen --built <name>.cbp --target=’Release’

<filename> 指定 CodeBlock 项目文件(*.cbp)或解决方案文件(*.workspace)。

--file=<filename>[:line]：使用 CodeBlocks 打开指定文件。可选的行号指示代码编辑器跳转到该行。

/h, --help：显示帮助信息。

/na, --no-check-associations：不执行文件关联检查。（windows only)

/nd, --no-dde：不启动 DDE 服务。(windows only)

/ni, --no-ipc：不启动 IPC 服务。(Linux and Mac only)

/ns, --no-splash-screen：应用程序启动的时候，不显示启动画面。

/d, --debug-log：显示应用程序的调试日志

--prefix=<str>：设置共享数据文件夹的前缀

/p, --personality=<str>, --profile=<str>：设置要使用的个性化配置。你可以使用“ask”参数来列

出可选择的个性化配置。

--rebuild：清理并重新编译工程或解决方案。

--build：编译工程或解决方案。

--target=<str>：设置编译模式，如：--target=’Release’

--no-batch-window-close：编译完成的时候，不关闭日志窗口。

--batch-build-notify：编译完成的时候显示提示信息

--safe-mode：启动的时候，所有插件都不可用。

> <build log file>：重定向标准输出到日志文件。这是标准 DOS/*nix shell 的输出重定向，并非 CodeBlocks 内置选项。

1.13 快捷键

在 IDE 中使用快捷键比使用鼠标更为高效。下表给出 CodeBolcks 默认的快捷键。（译者注：笔者将 CodeBlocks 中的快捷键设置为与 VS 大体一致，使用 CodeBlocks 时非常顺手。）

Function Shortcut Key

Undo last action Ctrl + Z

Redo last action Ctrl + Shift + Z

Cut selected text Ctrl + X

Copy selected text Ctrl + C

Paste text from clipboard Ctrl + V

Select all text Ctrl + A

Swap header / source F11

Comment highlighted code Ctrl + Shift + C

Uncomment highlighted code Ctrl + Shift + X

Duplicate line caret is on Ctrl + D

Auto-complete / Abbreviations Ctrl + Space / Ctrl + J

Show call tip Ctrl + Shift + Space

Swap line caret is on with line above it Ctrl + T

Toggle bookmark Ctrl + B

Goto previous bookmark Alt + PgUp

Goto next bookmark Alt + PgDown

Toggle current block folding F12

Toggle all folds Shift + F12

CodeBlocks 代码编辑器组件提供的快捷键，这些快捷键不能重新绑定（rebound）。

Function Shortcut Key

Magnify text size. Ctrl + Keypad "+"

Reduce text size. Ctrl + Keypad "-"

Restore text size to normal. Ctrl + Keypad "/"

Cycle through recent files. Ctrl + Tab

Indent block. Tab

Dedent block. Shift + Tab

Delete to start of word. Ctrl + BackSpace
Delete to end of word. Ctrl + Delete
Delete to start of line. Ctrl + Shift + BackSpace
Delete to end of line. Ctrl + Shift + Delete
Go to start of document. Ctrl + Home
Extend selection to start of document. Ctrl + Shift + Home
Go to start of display line. Alt + Home
Extend selection to start of display line. Alt + Shift + Home
Go to end of document. Ctrl + End
Extend selection to end of document. Ctrl + Shift + End
Go to end of display line. Alt + End
Extend selection to end of display line. Alt + Shift + End
Expand or contract a fold point. Ctrl + Keypad "*"
Create or delete a bookmark. Ctrl + F2
Go to next bookmark. F2
Select to next bookmark. Alt + F2
Find selection. Ctrl + F3
Find selection backwards. Ctrl + Shift + F3
Scroll up. Ctrl + Up
Scroll down. Ctrl + Down
Line cut. Ctrl + L
Line copy. Ctrl + Shift + T
Line delete. Ctrl + Shift + L
Line transpose with previous. Ctrl + T
Line duplicate. Ctrl + D
Find matching preprocessor conditional, skipping nested ones. Ctrl + K
Select to matching preprocessor conditional. Ctrl + Shift + K
Find matching preprocessor conditional backwards, skipping nested ones. Ctrl + J
Select to matching preprocessor conditional backwards. Ctrl + Shift + J
Previous paragraph. Shift extends selection. Ctrl + [
Next paragraph. Shift extends selection. Ctrl +]
Previous word. Shift extends selection. Ctrl + Left
Next word. Shift extends selection. Ctrl + Right
Previous word part. Shift extends selection. Ctrl + /
Next word part. Shift extends selection. Ctrl + \
Files
Function Shortcut Key
New file or project Ctrl + N
Open existing file or project Ctrl + O
Save current file Ctrl + S
Save all files Ctrl + Shift + S

Close current file Ctrl + F4 / Ctrl + W

Close all files Ctrl + Shift + F4 / Ctrl + Shift + W

CodeBlocks 的 Tab 组件所提供的快捷键，这些快捷键不能重新绑定（rebound）。

Function Shortcut Key

Activate next open file Ctrl + Tab

Activate previous open file Ctrl + Shift + Tab

View

Function Shortcut Key

Show / hide Messages pane F2

Show / hide Management pane Shift + F2

Move project up (in Project tree) Ctrl + Shift + Up

Move project down (in Project tree) Ctrl + Shift + Down

Activate prior (in Project tree) Alt + F5

Activate next (in Project tree) Alt + F6

Zoom in / out Ctrl + Roll Mouse Wheel

Focus editor CTRL + Alt + E

Search

Function Shortcut Key

Find Ctrl + F

Find next F3

Find previous Shift + F3

Find in files Crtl + Shift + F

Replace Ctrl + R

Replace in files Ctrl + Shift + R

Goto line Ctrl + G

Goto next changed line Ctrl + F3

Goto previous changed line Ctrl + Shift + F3

Goto file Alt + G

Goto function Ctrl + Alt + G

Goto previous function Ctrl + PgUp

Goto next function Ctrl + PgDn

Goto declaration Ctrl + Shift + .

Goto implementation Ctrl + .

Open include file Ctrl + Alt + .

Build

Function Shortcut Key

Build Ctrl + F9

Compile current file Ctrl + Shift + F9

Run Ctrl + F10

Build and Run F9

Rebuild Ctrl + F11

Debug

Function Shortcut Key

Debug F8

Continue debugging Ctrl + F7

Step over a code block F7

Step into a code block Shift + F7

Step out of a code block Ctrl + Shift + F7

Toggle breakpoint F5

Run to cursor F4

Previous error Alt + F1

Next error Alt + F2